PRG

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/458,883	12/10/1999	THOMAS R. PUZAK	YO999-589	9397

7590 08/12/2003

JAY P. SBROLLINI
IBM CORPORATION
INTELLECTUAL PROPERTY LAW DEPT
PO BOX 218
YORKTOWN, NY 10598

EXAMINER	
WOOD, WILLIAM H	
ART UNIT	PAPER NUMBER
2124	11

DATE MAILED: 08/12/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

MAILED
AUG 1 2 2003
Technology Center 2100

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Paper No. 11

Application Number: 09/458,883
Filing Date: December 10, 1999
Appellant(s): PUZAK ET AL.

Michael J. Buchenhorner
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 30 May 2003.

(1) Real Party in Interest

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

(3) Status of Claims

The statement of the status of the claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Invention

The summary of invention contained in the brief is correct.

(6) Issues

The appellant's statement of the issues in the brief is correct.

(7) Grouping of Claims

The rejection of claims 1, 3-12 and 14-22 stand or fall together.

(8) Claims Appealed

The copy of the appealed claims contained in the Appendix to the brief is correct.

(9) Prior Art of Record

5,742,804	Yeh et al.	4-1998

(10) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 3-12 and 14-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yeh et al. (USPN 5,742,804).

In regard to claim 1, Yeh taught the following limitations:

i) *a system including a high speed buffer logically placed between memory and at least one processor unit, a method of executing an instruction stream stored in the memory, wherein the instruction stream comprises a sequence of instructions including at least one prefetch instruction that prefetches information from the memory into the high speed buffer* (column 2, lines 27-34)

ii) *deriving first path data from a compiler by analyzing control flow information during compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction* (column 4, lines 8-23; column 6, lines 21-27; inserting this instruction indicates the first data had to be derived for the trace parameter of the instruction; the compiler indicates control flow information was analyzed)

iii) *obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction* (column 6, lines 28-37; specifically mentioned is dynamic branch prediction which indicates obtaining branch history of branches happening long before the prefetch is currently encountered)

iv) *generating second path data, wherein the second path data represents a predicted second path of execution* (column 6, lines 28-36; the execution path mentioned here is the second generated path data)

v) *determining whether the first path is consistent with the predicted second path* (column 6, lines 33-36)

vii) *prefetching instructions when the first path is consistent with the predicted second path* (column 6, lines 33-36)

Yeh did not explicitly state prefetching data. However, Yeh did allude to this concept of the prefetch instruction prefetching data to be operated on in column 1, lines 22-25. Here, Yeh states both instructions and data are needed to keep the computer running efficiently. Clearly if the instructions are being prefetched, the data for those instructions would need to be prefetched as well in order to keep up with the continual stream of operation. Furthermore, Yeh fetches data in that instructions contain explicit operands or the addresses to data additional data (the addresses are data). It would have been obvious to one of ordinary skill in the art to implement instructions with explicit data (i.e. a constant) or addresses to other data in Yeh. This would be obvious because this is essentially common practice for what memory access instructions do.

In regard to claim 3, Yeh taught the limitation *wherein the second path data is derived from information characterizing dynamic execution of the sequence of instructions by the processor unit* (column 6, lines 28-33; second path is the execution path).

In regard to claim 4, Yeh taught the limitation *wherein prefetch instruction is added to the instruction stream for execution by the at least one processor unit upon determining that the first path falls within the predicted second path* (column 6, lines 28-37; the unneeded prefetch requests are cancelled and in this manner only the prefetch instruction which is needed is added to the instruction stream for actual execution).

In regard to claim 5, Yeh taught the additional limitation *upon determining that the first path does not fall within the predicted second path, omitting the prefetch instruction from the instruction stream executed by the processor unit* (column 6, lines 28-37).

In regard to claim 6, Yeh taught the limitation *wherein the second path data are associated with one or more branch instructions* (column 6, lines 13-18; column 6, lines 28-33; Figure 1). Yeh does not explicitly state the limitation *wherein the second path data comprises a mask that represents a predicted path of execution that follows the associated branch instructions.* Though, in view of column 6, lines 7-20; column 6, lines 28-37, a masking pattern is being used for the first path, and one of ordinary skill in the art would recognize implementing Yeh's execution path (the second path) in the same bit pattern (i.e. "mask" indicating the path) would make the mentioned comparison operation easier. It would be a simple one-to-one comparison. Therefore, it would have been obvious to implement Yeh with a mask representing the path.

In regard to claim 7, Yeh taught the limitation *wherein the first path data comprises a mask that represents a path of execution from the prefetch instruction to the instruction that uses the information prefetched by the prefetch instruction* (column 4, lines 43-57; column 6, lines 7-20; the bit encodings represent the mask, which represents the path).

In regard to claim 8, Yeh taught the limitation *wherein the second path data are based upon accumulation of predictions associated with branch instructions* (column 6, lines 13-18; column 6, lines 28-33).

In regard to claim 9, Yeh taught the limitation *wherein the second path data is derived from predictions based upon previous execution of the instruction stream* (column 6, lines 13-18; column 6, lines 28-33).

In regard to claims 10 and 21, Yeh taught the additional limitation *wherein the prefetch instruction includes a field that identifies an instruction to prefetch from memory into the high speed buffer* (Figure 1, predict branch 6 instruction shown to have a "target" field; column 4, line 6).

In regard to claims 11 and 22, Yeh did not explicitly state limitation *wherein the prefetch instruction includes a field that identifies data to prefetch from memory into the high speed buffer, wherein the data is operated on by at least one instruction in the instruction stream*. However, Yeh did allude to this concept of the prefetch instruction prefetching data to be operated on in column 1, lines 22-25. Here, Yeh states both instructions and data are needed to keep the computer running efficiently. Clearly if the instructions are being prefetched, the data for those instructions would need to be prefetched as well in order to keep up with the continual stream of operation. Furthermore, Yeh fetches data in that instructions contain explicit operands or the addresses to data additional data (the addresses are data). It would have been obvious to one of ordinary skill in the art to implement instructions with explicit data (i.e. a constant) or addresses to other data in Yeh. This would be obvious because this is essentially common practice for what memory access instructions do. Therefore, it is obvious to implement Yeh as using a prefetch instruction to identify an instruction to prefetch which contains data that will be operated on, by at least one instruction.

In regard to claim 12, Yeh taught the following limitations

i) *a system including a memory storing an instruction stream comprising a sequence of instructions including at least one prefetch instruction, a processor unit for executing the sequence of instructions, and a high speed buffer logically placed between the memory and the one processor unit, an apparatus for conditionally executing the prefetch instruction* (column 2, lines 27-34)

ii) *decode logic for deriving first path data from a compiler performing static compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction* (column 4, lines 8-23; column 6, lines 21-27; inserting this instruction indicates the first data had to be derived for the trace parameter of the instruction; the compiler indicates control flow information was analyzed)

iii) *logic for obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction* (column 6, lines 28-37; specifically mentioned is dynamic branch prediction which indicates obtaining branch history of branches happening long before the prefetch is currently encountered)

iv) *path prediction logic for generating second path data, wherein the second path data represents a predicted second path of execution* (column 6, lines 28-36; the execution path mentioned here is the second generated path data)

v) *compare logic for determining whether the first path is consistent with the predicted second path* (column 6, lines 33-36)

vi) *execution logic for conditionally prefetching instructions when the first path is consistent with the predicted second path* (column 6, lines 33-36)

Yeh did not explicitly state prefetching data. However, Yeh did allude to this concept of the prefetch instruction prefetching data to be operated on in column 1, lines 22-25. Here, Yeh states both instructions and data are needed to keep the computer running efficiently. Clearly if the instructions are being prefetched, the data for those instructions would need to be prefetched as well in order to keep up with the continual stream of operation. Furthermore, Yeh fetches data in that instructions contain explicit operands or the addresses to data additional data (the addresses are data). It would have been obvious to one of ordinary skill in the art to implement instructions with explicit data (i.e. a constant) or addresses to other data in Yeh. This would be obvious because this is essentially common practice for what memory access instructions do.

In regard to claim 14, Yeh taught the limitation *wherein the second path data is derived from information characterizing dynamic execution of the sequence of instructions by the one processor unit* (column 6, lines 28-33; second path is the execution path).

In regard to claim 15, Yeh taught *wherein the execution logic executes the prefetch instruction upon determining that the first path falls within the predicted second path* (column 6, lines 28-37; the unneeded prefetch requests are cancelled and in this manner only the prefetch instruction which is actually needed is then executed).

In regard to claim 16, Yeh taught *wherein the execution logic omits execution of the prefetch instruction upon determining that the first path does not fall within the predicted second path* (column 6, lines 28-37).

In regard to claim 17, Yeh taught the additional limitation *wherein the second path data are associated with one or more branch instructions* (column 6, lines 13-18; column 6, lines 28-33; Figure 1). Yeh does not explicitly state *wherein the second path data comprises a mask that represents a predicted path of execution that follows the associated branch instructions.* Though, in view of column 6, lines 7-20; column 6, lines 28-37, a masking pattern is being used for the first path, and one of ordinary skill in the art would recognize implementing Yeh's execution path (the second path) in the same bit pattern (i.e. "mask" indicating the path) would make the mentioned comparison operation easier. It would be a simple one-to-one comparison. Therefore, it would have been obvious to implement Yeh with a mask representing the path.

In regard to claim 18, Yeh taught the limitation *wherein the first path data comprises a mask that represents path of execution from the prefetch instruction to the instruction that uses the information prefetched by the prefetch instruction* (column 4, lines 43-57; column 6, lines 7-20; the bit encodings represent the mask, which represents the path).

In regard to claim 19, Yeh taught the limitation *further comprising branch prediction logic for generating predictions associated with branch instructions, and a branch history queue for accumulating the predictions generated by the branch prediction logic, wherein the second path data generated by the path prediction logic is based upon the predictions accumulated in the branch history queue* (column 6, lines 13-18; column 6, lines 28-33). Of particular importance is the branch history queue, which is not explicitly stated in the above referenced passages, however it would have been obvious to one of ordinary skill that such a structure exists inherently in Yeh since some method of recording the branch histories is needed for prediction purposes.

In regard to claim 20, Yeh taught the limitation *wherein the predictions are based upon previous execution of the instruction stream* (column 6, lines 13-18; column 6, lines 28-33).

(11) Response to Argument

For the following reasons, it is believed that the above rejections should be sustained.

Appellant's arguments filed 30 May 2003 in regard to "The 'Obtaining a Path History' Element" have been fully considered but they are not persuasive. Appellant apparently argues: $^{1)}$ Yeh does not disclose Appelant's "branch history" (Brief: page 5, lines 9-11 and lines 17-20); and $^{2)}$ Yeh's branch prediction mechanism is performed in a different sequence than that of the claimed invention (Brief: page 4, lines 17-18; page 5, lines 13-14 and lines 26-27). Due to a fundamental lack of understanding concerning the art of *Branch Prediction*, Appellant is simply incorrect on the first issue. As pointed to in previous Office Actions column 6, lines 28-37 of Yeh disclosed obtaining a dynamic history of branches (execution path) for comparison to the statically produced trace vector. One of ordinary skill in the art, reading the above passage, would understand that <u>dynamic predictors</u> being used to provide an <u>execution path</u> is a branch history as claimed by Appellant. Those of ordinary skill in the art recognize dynamic prediction of an execution path involves recording results (or tabulating some prediction value) of branches of previous executions in order to predict the current execution path (i.e. which branches will be taken this time around and which won't; the branch history). Appellant has chosen to focus on differing terminology in Yeh and in doing so ignored the fact that Yeh does disclose branch history as would be apparent to those in the art of branch prediction. As to the second issue, Appellant argues at length about the sequence of determining or comparing the branch history, yet this sequencing is **nowhere** in the independent claims. The broadest reasonable interpretation of claim 1, for example, states: *obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction* (the branch history is derived from branch instructions that existed and were executed before a prefetch instruction); *determining whether the first path is consistent with the predicted second path* (comparing the compiler generated path with the dynamic path produced during execution); *and prefetching instructions and data when the first path is consistent with the predicted second path* (prefetching instructions and data, however this does **not** state avoiding prefetching instructions if the first and second path are not "consistent"; Yeh simply prefetches all with some being cancelled later). Appellant's arguments address issues that the claims don't support. As a final note, Yeh determines its branch history *prior* to encountering the prefetch instruction. Once again, one of ordinary skill in the art understands dynamic prediction of execution path is performed as branches are encountered (dynamically) through previous executions. Therefore, the new execution path prediction is already calculated before actually encountering the prefetch instruction. Appellant's arguments are beneath one of ordinary skill in the art concerning branch prediction and do not even represent an understanding of Appellant's own invention much less the cited prior art Yeh.

Appellant's arguments filed 30 May 2003 in regard to "Prefetching Instructions and Data" have been fully considered but they are not persuasive. Appellant evidently argues Yeh does not disclose prefetching data. It is true that Yeh did not *explicitly* state prefetching data. Yeh does state prefetching instructions (column 2, lines 27-30). One of ordinary skill in the art at the time of invention would understand that to prefetch an instruction means data is prefetched as well. An instruction is often composed of various elements, for example: ADD A1 B1. This is an instruction to perform the mathematical operation of adding element A1 to element B1. Further, an instruction could be ADD 100 C1, which would indicate adding 100 to C1. Data in these cases are A1, B1, C1 and 100, which are known as *operands* in the art of processor instructions. The value of an operand can be either explicitly stated such as 100 or use and address to locate the value. In either case, the explicit value or the address, the operands are data. Since instructions contain data and instructions are prefetched, it would be obvious to one of ordinary skill in the art at the time of invention to implement Yeh using common instructions (such as ADD), which would thus prefetch data. This very simple example, again demonstrates that Appellant's arguments and general understanding are beneath one of ordinary skill in the art. Appellant asserts the previous rejections are an "unsupported conclusion" and the Examiner and/or Board cannot make such statements (Brief: page 7, lines 13-19). However, Appellant merely alleges the rejection is error, without actually citing knowledge that would refute the rejection.

Appellant further argues the obviousness of the claimed invention over Yeh is in question (Brief: page 8). However, as stated above in the rejection and illustrated here, one of ordinary skill in the art would have found the claimed invention as a whole obvious over Yeh as a whole, for at least the reasons stated.

Appellant's arguments filed 30 May 2003 in regard to claim 4 have been fully considered but they are not persuasive. Here, Appellant attempts to make an extension of the "sequencing" argument found above for the independent claims. Appellant evidently believes *wherein the prefetch instruction is added to the instruction stream for execution by the at least one processor unit upon determining that the first path falls within the predicted second path* somehow adds the "sequencing" limitation. Yet, Appellant fails to even discuss the previous rejection of: column 6, lines 28-37; the unneeded prefetch requests are cancelled and in this manner only the prefetch instruction which is needed is added to the instruction stream for actual execution. The rejection basically provides for how the broadly claimed invention of Appellant reads upon Yeh. Canceling unneeded prefetch requests means unneeded prefetch requests are not executed (or not present in the instruction stream) to avoid *bottlenecking*. The final stream of instructions is produced in this manner or in other words this is the manner prefetches are *added* to the final instruction stream. Due to lack of argument against the rejection and the broadest reasonable interpretation of Appellant's claims, the claims read upon Yeh.

Appellant's arguments filed 30 May 2003 in regard to claim 6 have been fully considered but they are not persuasive. Appellant argues a second path mask would not have been obvious from the teachings of Yeh. However, the previous office actions made clear that in view of the first path's masking pattern (column 6, lines 7-20; column 6, lines 28-37). It would have been obvious to one of ordinary skill in the art implement a second path mask for the simple reason of one-to-one comparison, thus making comparing the two paths simple/easy/efficient. Appellant contests this straight forward use of the disclosed technology, yet offers no explanation as to why. Appellant offers not technical reason against such an obvious implementation, demonstrating no understanding of the cited prior art or the technical field of microprocessor operations.

Appellant's arguments filed 30 May 2003 in regard to claim 8 have been fully considered but they are not persuasive. Appellant argues once again the issues of claim 1 and the additional limitation of second path data based upon accumulation of predictions associated with branch instructions. Appellant does not understand the art of *branch prediction*, an art common to microprocessors. One of ordinary skill in the art understands dynamic prediction of execution path is performed as branches are encountered (dynamically or accumulation). The result of dynamic branch prediction is an ever more accurate prediction as more branches are encountered and dynamically accumulated into the overall result. Therefore, claim 8's rejection stands.

Appellant's arguments filed 30 May 2003 in regard to claims 11 and 22 have been fully considered but they are not persuasive. Appellant argues similarly as to the independent claims and as such is responded to above already.

In response to applicant's numerous assertions in the arguments that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971). It is believed that the cited prior art Yeh, the proper rejections cited above and the arguments presented here more than adequately demonstrate examiner did not use improper hindsight reasoning.

In conclusion, after careful consideration Examiner has found no evidence in the claimed invention or Appellant's arguments to otherwise warrant a rejection as stated based upon Yeh et al.

Respectfully submitted,

[illegible]

Todd Ingberg
Primary Examiner
Group 2100

William H. Wood
William H. Wood
July 30, 2003

Conferees
Kakali Chaki
Todd Ingberg

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

JAY P. SBROLLINI
IBM CORPORATION
INTELLECTUAL PROPERTY LAW DEPT
PO BOX 218
YORKTOWN, NY 10598